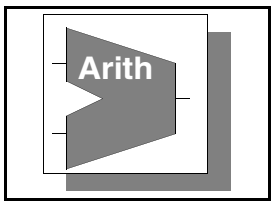
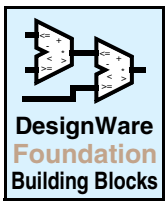




DW01_absval

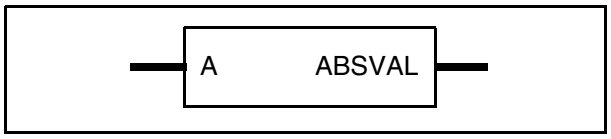
Absolute Value

Version, STAR, and myDesignWare Subscriptions: IP Directory

Features and Benefits

- Parameterized word length
- Inferable through a function call

Revision History



Description

DW01_absval determines the absolute value of data from input port A and places this value on output port ABSVAL.

The input is assumed to be represented as a signed two's complement number. The output is represented as an unsigned number.

Table 1-1 Pin Description

Pin Name	Width	Direction	Function
A	*width* bits	Input	Input data
ABSVAL	*width* bits	Output	Absolute value of A

Table 1-2 Parameter Description

Parameter	Values	Function
width	≥ 1	Word length of A and ABSVAL

Table 1-3 Synthesis Implementations[a]

Implementation Name	Function	License Feature Required
rpl	Ripple-carry synthesis model	none
cla	Carry-look-ahead synthesis model	none
clf	Fast carry-look-ahead synthesis model	DesignWare

a. During synthesis, Design Compiler selects the appropriate architecture for your constraints. However, you may force Design Compiler to use one of the architectures described in this table.

Table 1-4 Simulation Models

Model	Function
DW01.DW01_ABSVAL_CFG_SIM	Design unit name for VHDL simulation
dw/dw01/src/DW01_absval_sim.vhd	VHDL simulation model source code
dw/sim_ver/DW01_absval.v	Verilog simulation model source code

Table 1-5 Truth Table (*width* = 3)

A(2:0)	Value of A	ABSVAL(2:0)	Value of ABSVAL
000	0	000	0
001	1	001	1
010	2	010	2
011	3	011	3
100	–4	100	4[a]
101	–3	011	3
110	–2	010	2
111	–1	001	1

a. Only correct if the result is interpreted as an unsigned number.

Related Topics

- Math – Arithmetic Overview
- DesignWare Building Block IP User Guide

HDL Usage Through Function Inferencing - VHDL

```
library IEEE, DWARE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use DWARE.DW_foundation_arith.all;

entity DW01_absval_func is
  generic(width: integer:= 8);
  port( func_A : in std_logic_vector(width-1 downto 0);
        ABSVAL_func : out  std_logic_vector(width-1 downto 0) );
end DW01_absval_func;

architecture func of DW01_absval_func is
begin

  ABSVAL_func <= std_logic_vector(DWF_absval(SIGNED(func_A)));
end func;
```

HDL Usage Through Function Inferencing - Verilog

```
module DW01_absval_func (func_A, ABSVAL_func);
  parameter func_A_width = 8;

  // Passes the width to the absolute value function
  parameter width = func_A_width;

  // Please add search_path = search_path + {synopsys_root + "/dw/sim_ver"}
  // to your .synopsys_dc.setup file (for synthesis) and add
  // +incdir+$SYNOPSYS/dw/sim_ver+ to your verilog simulator command line
  // (for simulation).
  `include "DW01_absval_function.inc"

  input  [func_A_width-1 : 0] func_A;
  output [func_A_width-1 : 0] ABSVAL_func;
  assign ABSVAL_func = DWF_absval(func_A);

endmodule
```

HDL Usage Through Component Instantiation - VHDL

```
library IEEE,DWARE;
use IEEE.std_logic_1164.all;
use DWARE.DWpackages.all;
use DWARE.DW_foundation_comp.all;

entity DW01_absval_inst is
  generic ( inst_width : NATURAL := 8 );
  port ( inst_A : in std_logic_vector(inst_width-1 downto 0);
         ABSVAL_inst : out std_logic_vector(inst_width-1 downto 0) );
end DW01_absval_inst;

architecture inst of DW01_absval_inst is
begin

  -- Instance of DW01_absval
  U1 : DW01_absval
    generic map ( width => inst_width )
    port map ( A => inst_A, ABSVAL => ABSVAL_inst );
end inst;

-- pragma translate_off
configuration DW01_absval_inst_cfg_inst of DW01_absval_inst is
  for inst
  end for; -- inst
end DW01_absval_inst_cfg_inst;
-- pragma translate_on
```

HDL Usage Through Component Instantiation - Verilog

```
module DW01_absval_inst( inst_A, ABSVAL_inst );
  parameter width = 8;

  input [width-1 : 0] inst_A;
  output [width-1 : 0] ABSVAL_inst;

  // Instance of DW01_absval
  DW01_absval #(width)
    U1 ( .A(inst_A), .ABSVAL(ABSVAL_inst) );

endmodule
```

Revision History

For notes about this release, see the *DesignWare Building Block IP Release Notes*.

For lists of both known and fixed issues for this component, refer to the STAR report.

For a version of this datasheet with visible change bars, click here.

Date	Release	Updates
January 2019	DWBB_201806.5	■ Updated example in "HDL Usage Through Component Instantiation - VHDL" on page 4 ■ Added this Revision History table and the document links on this page

Copyright Notice and Proprietary Information

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Trademarks

Synopsys and certain Synopsys product names are trademarks of Synopsys, as set forth at https://www.synopsys.com/company/legal/trademarks-brands.html.

All other product or company names may be trademarks of their respective owners.

Free and Open-Source Software Licensing Notices

If applicable, Free and Open-Source Software (FOSS) licensing notices are available in the product installation.

Third-Party Links

Any links to third-party websites included in this document are for your convenience only. Synopsys does not endorse and is not responsible for such websites and their practices, including privacy practices, availability, and content.

Synopsys, Inc.
www.synopsys.com